1 Supplementary Material / Appendix for Graph Neural Networks for Soft Semi-Supervised Learning on Hypergraphs

The appendix is organised as follows:

1. Regularised Wasserstein and the Sinkhorn algorithm
2. Time complexity of the proposed DHN
3. Proofs and more notations used for theoretical analysis
4. Additional experiments (Graph-based soft SSL and Ablation Study)
5. Details of hyperparameters
6. Sources of real-world datasets

2 Regularised Wasserstein and the Sinkhorn Algorithm

The Wasserstein distance can be written as

$$\begin{aligned} L &= \sum_{v \in V_k} W_p\Big(Z_v, Y_v\Big) \\ W_p(\mu,\nu) &= \left(\inf_{\pi \in \Pi(\mu,\nu)} \int_{M \times M} C(x_1,x_2)^p d\pi(x_1,x_2) \right)^{\frac{1}{p}}. \end{aligned} \tag{1}$$

For discrete distributions, W_p of Equation 1 is the solution of a linear program:

$$\begin{aligned} W_p\left(z,y\right)^p &= \min \sum_{i=1}^{F} \sum_{j=1}^{F} C_{ij}^p \pi_{ij} \\ \text{s.t.} \quad \pi_{ij} &\geq 0, \sum_{j=1}^{F} \pi_{ij} = [z]_i\,, \sum_{i=1}^{F} \pi_{ij} = [y]_j \qquad \forall 1 \leq i,j \leq F \end{aligned} \tag{2}$$

where for an arbitrary element $q \in \mathcal{P}_F$, $[q]_i$ stands for the probability mass in the ith bin.

2.1 Sinkhorn divergence

The expensive linear program 2 can be efficiently solved by entropic regularisation [3]:

$$W_p^\lambda(z,y)^p = \min \sum_{i=1}^{F} \sum_{j=1}^{F} C_{ij}^p \pi_{ij} + \lambda \cdot \mathrm{Tr}\Big(\pi\big(\log \pi - \mathbf{1}\mathbf{1}^T\big)^T\Big) \tag{3}$$

$$\text{s.t.} \quad \pi_{ij} \geq 0, \sum_{j=1}^{F} \pi_{ij} = [z]_i\,, \sum_{i=1}^{F} \pi_{ij} = [y]_j \qquad \forall 1 \leq i,j \leq F$$

where $\log(\cdot)$ is applied element-wise and $\lambda \geq 0$ is a hyperparameter. The optimal solution π^* for $\lambda > 0$ takes the following form.

$$\pi^* = \text{diag}(r) \cdot \exp\left(- \frac{C^p}{\lambda}\right) \cdot \text{diag}(c)$$

where $\text{diag}(\mathbf{z})$ is a diagonal matrix with the components of $\mathbf{z}$ in the diagonal places.

Sinkhorn algorithm [3]: We optimise Equation 3 for r and c via matrix balancing, i.e., start with an initial $K := \exp(-\frac{C^p}{\lambda})$ and alternately ensure the marginal constraints are satisfied until convergence:

$$r \leftarrow z./(Kc) \qquad c \leftarrow y./(K^T r)$$

where ./ is element-wise division. We use the above efficient algorithm in our experiments.

3 Time complexity of DHN

We consider the problem of predicting probability distributions for the vertices in $\mathcal{H} = (V, E_d)$ given a typically small subset $V_k \subseteq V$ of vertices with known distributions. In this work, we are concerned with discrete distributions modelled on a complete separable metric space (M, C). Furthermore, we assume that we are provided with a feature matrix, $X_V \in \mathbb{R}^{n \times D_V}$, in which each vertex $v \in V$ is represented by a D_v-dimensional feature vector x_v (here $n = |V|$). We are also provided with a hyperedge feature matrix $X_E \in \mathbb{R}^{m \times D_E}$ with $x_e, e \in E$ as D_E-dimensional feature representations. Let T be the time taken for the Sinkhorn algorithm on all the vertices with known distributions. Note that T can be approximated in near-linear time [1]. Further, let T be the total number of epochs of training. Define

$$N := \sum_{e \in E} |e|, \qquad N_c := \sum_{e \in E} {}^{|e|}C_2$$

The time complexity oa one-layer DHN is $O\Big(|E_d| \cdot D_E \cdot h1 + |E| \cdot D_V \cdot h_2\Big)$ where h_1 is the number of hidden units of the GNN layer and h_2 is the number of output channels.

4 Theoretical analysis: Generalisation error

A key contribution of this chapter is to provide generalisation error bounds for (hyper)graph-based soft-SSL. Our effort to derive these bounds has lead us to a more powerful outcome, namely, proving generalisation error bounds for a one-layer GNN by extending the results of a traditional GCN [12] to the soft SSL setting with Wasserstein loss. The main novelty is to generalise the error bounds to the learning problem "valued in the Wasserstein space." The main challenge is that the Wasserstein space is an abstract metric space without linear structure.

The section is organised as follows. We first introduce all the notations needed (ego-graph view, semi-supervised learning setting, etc.). We then give single layer and SGD bounds using the notations. We finally give the main result (proposition 1) which states that a GNN trained with Wasserstein loss has the same generalisation error bound as the traditional GCN (trained with cross entropy).

Let $G = (V, E)$ be a connected graph with $|V| = n$ vertices. We consider GNN of a single layer

$$f\left(X, \Theta\right) = \sigma\left(KX\Theta\right) \tag{4}$$

where $X \in \mathbb{R}^{n\times d}$ is the feature matrix (n is the number of vertices in a graph, d is the dimension of the feature vectors), $K = g\left(L_G\right)$ is a graph filter (typically symmetrically normalised adjacency with self loops, and $L_G \in \mathbb{R}^{n\times n}$ is the graph Laplacian), and $\Theta \in \mathbb{R}^{d\times F}$ is the set of parameters. We note that our proposed DHN falls under this formulation in special circumstances. Specifically if the non-linearity σ_1 in is removed we get the kernel $K = II^T$ (also known as the clique expansion of the hypergraph [16].) The non-linearity σ in Equation 4 is the softmax function acting on each row of the product $KX\Theta \in \mathbb{R}^{n\times F}$; the output is of dimension $n \times F$, where each output row is a discrete probability distribution, i.e.,

$$f\left(X, \Theta\right) \geq 0 \quad \text{and} \quad f\left(X, \Theta\right)\mathbf{1}_F = \mathbf{1}_n$$

where $\mathbf{1}_F = \left(1, \ldots, 1\right)^F \in \mathbb{R}^F$, and similarly for $\mathbf{1}_n$. Without loss of generality, we assume $d = 1$. Note that in order for the output to be nontrivial probability distributions, we must assume $F > 1$.

We adopt an ego-graph view [12] to simplify our discussion for local behavior of the soft GCN at a particular vertex. Whenever no confusion arises, we identify a vertices x and χ in the graph G with their respective D-dimensional feature vectors. Thus the output of f at $x \in V$ is

$$\begin{aligned} f\left(x, \Theta\right) &= \sigma\left(\sum_{\chi\in\mathcal{N}(x)} K_{x\chi}\chi\Theta\right) \\ &= \sigma\left(\left(\sum_{\chi\in\mathcal{N}(x)} K_{x\chi}\chi\right)\cdot\Theta\right) \end{aligned}$$

where $\mathcal{N}\left(x\right)$ denotes for the one-hop neighborhood of x with respect to the adjacency relation defined by matrix K, and $K_{x\chi} \in \mathbb{R}$ stands for the entry in $K \in \mathbb{R}^{n\times n}$ that describes the adjacency relation between vertices x and χ. Let $E_x := \sum_{\chi\in\mathcal{N}(x)} K_{x\chi}\chi \in \mathbb{R}$ so that $f\left(x, \Theta\right) = \sigma\left(E_x \cdot \Theta\right)$.

We consider the supervised learning setting, and learn GNN from the training set $\{z_i = (x_i, y_i), i = 1, \ldots, m\}$ sampled i.i.d. from the product space $V \times \mathcal{P}_F$ with respect to probability distribution $\mathcal{D}$ on this product space, where $\mathcal{P}_F$ is the space of discrete probability distributions on F atoms. The output of softmax lies in $\mathcal{P}_F$, which is a convex cone. For any new data $z = (x, y) \sim \mathcal{D}$, we evaluate the performance of GNN f using a Wasserstein cost

$$\ell\left(f\left(\cdot, \Theta\right), z\right) = \ell\left(f\left(\cdot, \Theta\right), (x, y)\right) = W\left(f\left(x, \Theta\right), y\right).$$

Here the Wasserstein cost is defined with respect to a cost function penalizing moving masses across bins. Since we are working only with histograms in GNN, we shall use a cost function $C \in \mathbb{R}^{F\times F}$ that is defined for pairs of histogram bins. The transport problem is a linear program with $z = f(x, \Theta)$.

4.1 Assumptions/Notations

To avoid unnecessary technical complications, assume the histogram admits a geometric realisation over the one-dimensional Euclidean space, such that the ith bin is placed at location $b_i \in \mathbb{R}$, and set

$$C_{ij} := \left|b_i - b_j\right|, \quad \forall 1 \le i, j \le F.$$

Without loss of generality, we assume $b_1 \le b_2 \le \cdots \le b_F$, and write $h_i := b_{i+1} - b_i \ge 0$ for all $i = 1, \ldots, F-1$. Denote the diameter of the support by $D := \max_{1\le i,j\le F} |b_i - b_j| = b_F - b_1$. We take the Wasserstein cost as the Wasserstein-1 distance: $W\left(\mu, \nu\right) := W_1\left(\mu, \nu\right)$. In this particular one-dimensional setting, we have a particularly simple form for the cost function:

$$W_1\left(\mu, \nu\right) = \int_0^1 \left|F_\mu^{-1}\left(s\right) - F_\nu^{-1}\left(s\right)\right| \mathrm{d}s = \int_{-\infty}^{\infty} \left|F_\mu\left(t\right) - F_\nu\left(t\right)\right| \mathrm{d}t \qquad (5)$$

where $F_\mu : \mathbb{R} \to [0, 1]$, $F_\nu : \mathbb{R} \to [0, 1]$ are the cumulative distribution functions of μ, ν, respectively; F_μ^{-1}, F_ν^{-1} are the *generalized inverses* of F_μ and F_ν, respectively, defined as (similar for F_ν^{-1})

$$F_\mu^{-1}\left(t\right) := \inf\left\{b \in \mathbb{R} : F_\mu\left(b\right) > t\right\}, \qquad \forall t \in \left[0, 1\right]. \qquad (6)$$

This characterisation is seen in any standard literature on optimal transport, e.g., [13].

4.2 Definitions: Generalisation and Empirical Erros

Let the learning algorithm A_S on a dataset S be a function from ζ^m to $(\mathcal{Y})^X$. where $\mathcal{X}$ is the input Hilber space, $\mathcal{Y}$ is the output Hilbert space, $\zeta = \mathcal{X} \times \mathcal{Y}$. The training set of datapoints, labels is $S = \{z_1 = (x_1, y_1), \cdots, z_k = (x_k, y_k))\}$. Let the loss function be $\ell : \zeta^m \times \zeta \to \mathbb{R}$. Then the generalisation error or risk $R(A_S)$ is deined as

$$R(A_S) := \mathbb{E}\Big[\ell(A_S, z)\Big] = \int \ell(A_S, z)p(z)dz$$

where $p(z)$ is the probability of seeing the sample $z \in S$.

The empirical error, on the other hand, is defined as

$$R_{\mathrm{emp}}(A_S) := \frac{1}{k}\sum_{j=1}^{k} \ell(A_S, z_j)$$

4.3 Algorithmic Stability

Denote $S \subset V \times \mathcal{P}_F$ for an arbitrary training data set sampled with respect to distribution $\mathcal{D}$. A learning algorithm for GNN, denoted as A, maps a training set S to a trained GNN $f(\cdot, \Theta_S) : x \mapsto f(x, \Theta_S)$. Let S' be another training data set that differs from S by exactly one data. Our goal in this section is to establish a uniform stability

$$\begin{aligned}
&\sup_{\substack{S \subset V \times \mathcal{P}_F \\ (x,y) \in V \times \mathcal{P}_F}} \left|\mathbb{E}_A\left[\ell\left(f\left(\cdot, \Theta_S\right), (x,y)\right)\right] - \mathbb{E}_A\left[\ell\left(f\left(\cdot, \Theta_{S'}\right), (x,y)\right)\right]\right| \\
&= \sup_{\substack{S \subset V \times \mathcal{P}_F \\ (x,y) \in V \times \mathcal{P}_F}} \left|\mathbb{E}_A\left[W_1\left(f\left(x, \Theta_S\right), y\right)\right] - \mathbb{E}_A\left[W_1\left(f\left(x, \Theta_{S'}\right), y\right)\right]\right| \\
&= \sup_{\substack{S \subset V \times \mathcal{P}_F \\ (x,y) \in V \times \mathcal{P}_F}} \left|\mathbb{E}_A\left[W_1\left(f\left(x, \Theta_S\right), y\right) - W_1\left(f\left(x, \Theta_{S'}\right), y\right)\right]\right| \leq 2\beta_m \qquad (7)
\end{aligned}$$

with $\beta_m = O(1/m)$, which in turn can be used to establish the generalisation error bound of the form [12], following the framework [5] for SGD-based learning algorithms. Following the arguments in [12], this boils down to checking a few Lipschitz properties for the cost function.

We now derive single layer and SGD bounds.

4.4 Extension to GNNs on hypergraphs

We note that our proposed DHN falls under this formulation in special circumstances. In particular, if the non-linearities are removed, our DHN can be seen as $K = IA^2$ where $f_{GNN}(\mathcal{G}, X_E)$ is the simple graph convolution operator [15] and A is the symmetrically normalised adjacency (with self loops) of the graph $\mathcal{G}$. Also, the analysis can be extended to exisiting hypergraph GNNs such as hypergraph neural network (HGNN [4]) where $K = II^T$ (also known as the clique expansion [16] of the hypergraph). Hence, our theoretical analysis is quite general that can be easily appplied to DHN and existing hypergraph neural methods.

4.5 Single layer bound

By the triangle inequality,

$$\left|W_1\left(f\left(x, \Theta_S\right), y\right) - W_1\left(f\left(x, \Theta_{S'}\right), y\right)\right| \leq W_1\left(f\left(x, \Theta_S\right), f\left(x, \Theta_{S'}\right)\right).$$

By [14], recall that the diameter of the support is D, we have

$$W_1\left(f\left(x,\Theta_S\right),f\left(x,\Theta_{S'}\right)\right)\leq D\left\|f\left(x,\Theta_S\right)-f\left(x,\Theta_{S'}\right)\right\|_{\mathrm{TV}}$$

where $\left\|\cdot\right\|_{\mathrm{TV}}$ is the total variation distance, which by definition is

$$\begin{aligned}\left\|f\left(x,\Theta_S\right)-f\left(x,\Theta_{S'}\right)\right\|_{\mathrm{TV}}&=\frac{1}{2}\sum_{i=1}^{F}\left|\left[f\left(x,\Theta_S\right)\right]_i-\left[f\left(x,\Theta_{S'}\right)\right]_i\right|\\&=\frac{1}{2}\left\|\sigma\left(E_x\cdot\Theta_S\right)-\sigma\left(E_x\cdot\Theta_{S'}\right)\right\|_1\end{aligned}$$

where $\left\|\cdot\right\|_1$ is the L^1-distance on $\mathbb{R}^F$. Since the softmax function is Lipschitz continuous, we have

$$\left\|f\left(x,\Theta_S\right)-f\left(x,\Theta_{S'}\right)\right\|_{\mathrm{TV}}\leq\frac{L_\sigma}{2}\left|E_x\right|\cdot\left\|\Theta_S-\Theta_{S'}\right\|_1$$

and thus

$$\left|\mathbb{E}_A\left[W_1\left(f\left(x,\Theta_S\right),y\right)-W_1\left(f\left(x,\Theta_{S'}\right),y\right)\right]\right| \tag{8}$$

$$\leq\frac{L_\sigma D}{2}\sup_{x\in V}\left|E_x\right|\cdot\mathbb{E}_A\left\|\Theta_S-\Theta_{S'}\right\|_1=\frac{L_\sigma D}{2}g_\lambda\mathbb{E}_A\left\|\Theta_S-\Theta_{S'}\right\|_1 \tag{9}$$

where we used notation $g_\lambda:=\sup_{x\in V}\left|E_x\right|$, which is known to be upper bounded by $\lambda_G^{\max}$, the spectrum of the graph Laplacian L_G with largest absolute value [12].

4.6 SGD bound

It now remains to bound $\mathbb{E}_A\left\|\Theta_S-\Theta_{S'}\right\|_1$ resulting from the SGD iterations. The main technical challenge, as noted before, is to generalise the results to the Wasserstein space which is an abstract metric space without linear structure. Specifically, we have to modify the "gradient" in the Wasserstein space as the straightforward version [12] does not satisfy the Lipschitz condition required in the algorithmic stability framework. To the best of our knowledge this modification is not seen in existing literature and can be thought of as a generalisation of the "gradient clipping" operation [5].

4.7 SGD Bound: Proof

Note that

$$W_1\left(\mu,\nu\right)=\int_0^1\left|F_\mu^{-1}\left(s\right)-F_\nu^{-1}\left(s\right)\right|\mathrm{d}s=\int_{-\infty}^{\infty}\left|F_\mu\left(t\right)-F_\nu\left(t\right)\right|\mathrm{d}t \tag{10}$$

and from the single-layer bound

$$\left|\mathbb{E}_A\left[W_1\left(f\left(x,\Theta_S\right),y\right)-W_1\left(f\left(x,\Theta_{S'}\right),y\right)\right]\right| \le \frac{L_\sigma D}{2}\sup_{x\in V}\left|E_x\right|\cdot\mathbb{E}_A\left\|\Theta_S-\Theta_{S'}\right\|_1 = \frac{L_\sigma D}{2} g_\lambda \mathbb{E}_A\left\|\Theta_S-\Theta_{S'}\right\|_1 \tag{11}$$

Uniform stability:

$$\begin{aligned}
&\sup_{\substack{S\subset V\times\mathcal{P}_F\\(x,y)\in V\times\mathcal{P}_F}}\left|\mathbb{E}_A\left[\ell\left(f\left(\cdot,\Theta_S\right),(x,y)\right)\right]-\mathbb{E}_A\left[\ell\left(f\left(\cdot,\Theta_{S'}\right),(x,y)\right)\right]\right|\\
&=\sup_{\substack{S\subset V\times\mathcal{P}_F\\(x,y)\in V\times\mathcal{P}_F}}\left|\mathbb{E}_A\left[W_1\left(f\left(x,\Theta_S\right),y\right)\right]-\mathbb{E}_A\left[W_1\left(f\left(x,\Theta_{S'}\right),y\right)\right]\right|\\
&=\sup_{\substack{S\subset V\times\mathcal{P}_F\\(x,y)\in V\times\mathcal{P}_F}}\left|\mathbb{E}_A\left[W_1\left(f\left(x,\Theta_S\right),y\right)-W_1\left(f\left(x,\Theta_{S'}\right),y\right)\right]\right|\le 2\beta_m
\end{aligned} \tag{12}$$

It now remains to bound $\mathbb{E}_A\left\|\Theta_S-\Theta_{S'}\right\|_1$ resulting from the SGD iterations. Given training set S, applying SGD to GNN amounts to performing the updates

$$\Theta_{S,t+1}=\Theta_{S,t}-\eta\nabla_\Theta\ell\left(f\left(\cdot,\Theta\right),\left(x_{i_t},y_{i_t}\right)\right)=\Theta_{S,t}-\eta\nabla_\Theta W_1\left(f\left(x_{i_t},\Theta_{S,t}\right),y_{i_t}\right)$$

where $\eta>0$ is the learning rate and $z_{i_t}=\left(x_{i_t},y_{i_t}\right)$ are random data i.i.d. uniformly sampled from the training set. By the simple formulae (10) for one-dimensional optimal transport, we can explicitly write out for any parameter set Θ and data $z=(x,y)$

$$\begin{aligned}
W_1\left(f\left(x,\Theta\right),y\right)&=\int_{-\infty}^{\infty}\left|F_{f(x,\Theta)}\left(t\right)-F_y\left(t\right)\right|\,\mathrm{d}t=\sum_{i=1}^{F-1}\left(x_{i+1}-x_i\right)\left|\sum_{j=1}^{i}\left(\left[f\left(x,\Theta\right)\right]_i-[y]_i\right)\right|\\
&=\sum_{i=1}^{F-1}h_i\left|\sum_{j=1}^{i}\left(\left[\sigma\left(E_x\cdot\Theta\right)\right]_i-[y]_i\right)\right|
\end{aligned}$$

where again we used notation $[y]_i$ to denote the probability mass of $y\in\mathcal{P}_F$ in the ith bin, for all $i=1,\ldots,F$. Thus

$$\begin{aligned}
&\frac{\partial}{\partial \Theta_k} W_1\left(f\left(x,\Theta\right),y\right) = \sum_{i=1}^{F-1} h_i \cdot \operatorname{sgn}\left\{\sum_{j=1}^{i}\left(\left[\sigma\left(E_x\cdot\Theta\right)\right]_j - [y]_j\right)\right\} \cdot \sum_{j=1}^{i} \frac{\partial}{\partial\Theta_k}\left[\sigma\left(E_x\cdot\Theta\right)\right]_j \\
&\quad = \sum_{i=1}^{F-1} h_i \cdot \operatorname{sgn}\left\{\sum_{j=1}^{i}\left(\left[\sigma\left(E_x\cdot\Theta\right)\right]_j - [y]_j\right)\right\} \cdot E_x \sum_{j=1}^{i}\left[\sigma\left(E_x\cdot\Theta\right)\right]_j\left(\delta_{jk} - \left[\sigma\left(E_x\cdot\Theta\right)\right]_k\right) \\
&\quad = E_x\cdot\left[\sigma\left(E_x\cdot\Theta\right)\right]_k \sum_{i=1}^{F-1} h_i \cdot \operatorname{sgn}\left\{\sum_{j=1}^{i}\left(\left[\sigma\left(E_x\cdot\Theta\right)\right]_j - [y]_j\right)\right\}\left(1 - \sum_{j=i}^{j}\left[\sigma\left(E_x\cdot\Theta\right)\right]_j\right) \\
&\quad = E_x\cdot\left[\sigma\left(E_x\cdot\Theta\right)\right]_k \sum_{i=1}^{F-1} h_i \cdot \operatorname{sgn}\left\{\sum_{j=1}^{i}\left(\left[\sigma\left(E_x\cdot\Theta\right)\right]_j - [y]_j\right)\right\}\left(\sum_{j=i}^{j}\left([y]_j - \left[\sigma\left(E_x\cdot\Theta\right)\right]_j\right) + \sum_{j=i+1}^{F}[y]_j\right) \\
&\quad = -E_x\cdot\left[\sigma\left(E_x\cdot\Theta\right)\right]_k \cdot W_1\left(\sigma\left(E_x\cdot\Theta\right),y\right) \\
&\quad +E_x\cdot\left[\sigma\left(E_x\cdot\Theta\right)\right]_k \sum_{i=1}^{F-1} h_i \cdot \operatorname{sgn}\left\{\sum_{j=1}^{i}\left(\left[\sigma\left(E_x\cdot\Theta\right)\right]_j - [y]_j\right)\right\}\sum_{j=i+1}^{F}[y]_j
\end{aligned}$$

where $\operatorname{sgn}\{\cdot\}$ is the sign function, and δ_{ik} is the Kronecker delta notation. The second equality used the specific form of the derivative of the softmax function. Unfortunately, this gradient is not Lipschitz continuous due to the sign function in the second term. Nevertheless, if we use a "modified gradient" that drops the second term, i.e., choose to update the parameter Θ_k in the direction

$$\frac{\widetilde{\partial}}{\partial\Theta_k} W_1\left(f\left(x,\Theta\right),y\right) := -E_x\cdot\left[\sigma\left(E_x\cdot\Theta\right)\right]_k\cdot W_1\left(\sigma\left(E_x\cdot\Theta\right),y\right) \tag{13}$$

then obviously this new choice of "descent" direction is certainly Lipschitz continuous, as

$$\begin{aligned}
&\left|\frac{\widetilde{\partial}}{\partial\Theta_k} W_1\left(f\left(x,\Theta_S\right),y\right) - \frac{\widetilde{\partial}}{\partial\Theta_k} W_1\left(f\left(x,\Theta_{S'}\right),y\right)\right| \\
&= \left|E_x\right|\cdot\left|\left[\sigma\left(E_x\cdot\Theta_S\right)\right]_k\cdot W_1\left(\sigma\left(E_x\cdot\Theta_S\right),y\right) - \left[\sigma\left(E_x\cdot\Theta_{S'}\right)\right]_k\cdot W_1\left(\sigma\left(E_x\cdot\Theta_{S'}\right),y\right)\right| \\
&\le \left|E_x\right|\cdot\Bigg[\left|\left[\sigma\left(E_x\cdot\Theta_S\right)\right]_k - \left[\sigma\left(E_x\cdot\Theta_{S'}\right)\right]_k\right|\cdot W_1\left(\sigma\left(E_x\cdot\Theta_S\right),y\right) \\
&\qquad + \left[\sigma\left(E_x\cdot\Theta_{S'}\right)\right]_k\cdot\left|W_1\left(\sigma\left(E_x\cdot\Theta_S\right),y\right) - W_1\left(\sigma\left(E_x\cdot\Theta_{S'}\right),y\right)\right|\Bigg] \\
&\le \left|E_x\right|\cdot\left(L_\sigma D\left|E_x\right|\cdot\left\|\Theta_S-\Theta_{S'}\right\|_1 + W_1\left(\sigma\left(E_x\cdot\Theta_S\right),\sigma\left(E_x\cdot\Theta_{S'}\right)\right)\right) \\
&\le \left|E_x\right|\cdot\frac{3}{2}L_\sigma D\cdot\left|E_x\right|\cdot\left\|\Theta_S-\Theta_{S'}\right\|_1 \le \frac{3}{2}L_\sigma D\cdot g_\lambda^2\left\|\Theta_S-\Theta_{S'}\right\|_1.
\end{aligned}$$

Therefore, if we define

$$\widetilde{\nabla}_\Theta W_1\left(f\left(x,\Theta\right),y\right) := \left(\widetilde{\frac{\partial}{\partial \Theta_1}} W_1\left(f\left(x,\Theta\right),y\right), \cdots, \widetilde{\frac{\partial}{\partial \Theta_F}} W_1\left(f\left(x,\Theta\right),y\right)\right)^\top \tag{14}$$

then the stochastic update algorithm[1]

$$\Theta_{S,t+1} = \Theta_{S,t} - \eta \widetilde{\nabla}_\Theta W_1\left(f\left(x_{i_t},\Theta_{S,t}\right),y_{i_t}\right) \tag{15}$$

will still satisfy the generalization bound, due to the Lipschitz continuity

$$\left\|\widetilde{\nabla}_\Theta W_1\left(f\left(x,\Theta_S\right),y\right) - \widetilde{\nabla}_\Theta W_1\left(f\left(x,\Theta_{S'}\right),y\right)\right\|_1 \le \frac{3D}{2} L_\sigma g_\lambda^2 \left\|\Theta_S - \Theta_{S'}\right\|_1 . \tag{16}$$

In fact, (16) is exactly the GNN analogy of [12] which establishes the stability for the "same sample loss" case. The "different sample loss" analogy, or the lemma in [12], can be trivially obtained by the definition (13). In fact, noting that

$$\left|\widetilde{\frac{\partial}{\partial \Theta_k}} W_1\left(f\left(x,\Theta\right),y\right)\right| = \left|E_x \cdot \left[\sigma\left(E_x \cdot \Theta\right)\right]_k \cdot W_1\left(\sigma\left(E_x \cdot \Theta\right),y\right)\right| \le g_\lambda D,$$

we easily obtain

$$\left|\widetilde{\frac{\partial}{\partial \Theta_k}} W_1\left(f\left(x,\Theta_S\right),y\right) - \widetilde{\frac{\partial}{\partial \Theta_k}} W_1\left(f\left(x',\Theta_{S'}\right),y'\right)\right| \le 2 g_\lambda D$$

and it follows that

$$\left\|\widetilde{\nabla}_\Theta W_1\left(f\left(x,\Theta_S\right),y\right) - \widetilde{\nabla}_\Theta W_1\left(f\left(x,\Theta_{S'}\right),y\right)\right\|_1 \le 2 F g_\lambda D. \tag{17}$$

Putting together (16) and (17), we obtain the following analogy of [12]: Starting with two training data sets S, S' that differs by exactly one sample, after each iteration t we have

[1] Note that this is not even a stochastic gradient descent algorithm! the "gradient" involved is a "fake" gradient — this is the counter-intuitive part.

$$
\begin{aligned}
&\mathbb{E}_A\left[\|\Theta_{S,t+1}-\Theta_{S',t+1}\|_1\right] \\
&=\mathbb{E}_A\left[\left\|\Theta_{S,t}-\eta\widetilde{\nabla}_\Theta W_1\left(f\left(x_t,\Theta_{S,t}\right),y_t\right)-\Theta_{S',t}+\eta\widetilde{\nabla}_\Theta W_1\left(f\left(x_t',\Theta_{S',t}\right),y_t'\right)\right\|\right] \\
&\leq \mathbb{E}_A\left[\|\Theta_{S,t+1}-\Theta_{S',t+1}\|_1\right]+\left(1-\frac{1}{m}\right)\cdot\eta\cdot\frac{3D}{2}L_\sigma g_\lambda^2\mathbb{E}_A\left[\|\Theta_{S,t}-\Theta_{S',t}\|_1\right]+\frac{1}{m}\cdot\eta\cdot 2Fg_\lambda D \\
&\leq\left(1+\frac{3}{2}\eta DL_\sigma g_\lambda^2\right)\mathbb{E}_A\left[\|\Theta_{S,t}-\Theta_{S',t}\|_1\right]+\frac{2\eta Fg_\lambda D}{m}.
\end{aligned}
$$

Solving this first-order recursion gives the stability after T random update steps:

$$
\mathbb{E}_A\left[\|\Theta_{S,T}-\Theta_{S',T}\|_1\right]\leq\frac{2\eta Fg_\lambda D}{m}\sum_{t=1}^{T}\left(1+\frac{3}{2}\eta DL_\sigma g_\lambda^2\right)^{t-1}. \tag{18}
$$

Combining (11) and (21) gives us

$$
\left|\mathbb{E}_A\left[W_1\left(f\left(x,\Theta_S\right),y\right)-W_1\left(f\left(x,\Theta_{S'}\right),y\right)\right]\right|\leq\frac{\eta FL_\sigma g_\lambda^2D^2}{m}\sum_{t=1}^{T}\left(1+\frac{3}{2}\eta DL_\sigma g_\lambda^2\right)^{t-1}. \tag{19}
$$

Therefore, we actually have the uniform algorithmic stability (12) holds with

$$
\beta_m=\frac{\eta FL_\sigma g_\lambda^2D^2}{2m}\sum_{t=1}^{T}\left(1+\frac{3}{2}\eta DL_\sigma g_\lambda^2\right)^{t-1}. \tag{20}
$$

$$
\mathbb{E}_A\left[\|\Theta_{S,T}-\Theta_{S',T}\|_1\right]\leq\frac{2\eta Fg_\lambda D}{m}\sum_{t=1}^{T}\left(1+\frac{3}{2}\eta DL_\sigma g_\lambda^2\right)^{t-1}. \tag{21}
$$

Combining (8) and (21) gives us

$$
\left|\mathbb{E}_A\left[W_1\left(f\left(x,\Theta_S\right),y\right)-W_1\left(f\left(x,\Theta_{S'}\right),y\right)\right]\right| \tag{22}
$$

$$
\leq\frac{\eta FL_\sigma g_\lambda^2D^2}{m}\sum_{t=1}^{T}\left(1+\frac{3}{2}\eta DL_\sigma g_\lambda^2\right)^{t-1}. \tag{23}
$$

Therefore, we actually have the uniform algorithmic stability holds with

$$
\beta_m=\frac{\eta FL_\sigma g_\lambda^2D^2}{2m}\sum_{t=1}^{T}\left(1+\frac{3}{2}\eta DL_\sigma g_\lambda^2\right)^{t-1}. \tag{24}
$$

Note that here $\beta_m=O(\frac{1}{m})$ (needed to obtain a tight generalisation bound).

4.8 Putting everything together

Lemma 1. *[12]: Let λ_G^{max} be the maximum absolute eigenvalue of L_G. Let G_x be the ego-graph of a vertex $x \in V$ with corresponding maximum absolute eigenvalue $\lambda_{G_x}^{max}$. Then the following eigenvalue (singular value) bound holds $\forall x \in V$,*

$$\lambda_{G_x}^{max} \leq \lambda_G^{max} \tag{25}$$

Lemma 2. *[12] A uniform stable randomised algorithm (A_S, β_m) with a bounded loss function $0 \leq \ell(A_S, \mathbf{y}) \leq B$, satisfies the following generalisation bound with probability at least $1-\delta$, over the random draw of $S, \mathbf{y}$ with $\delta \in (0,1)$,*

$$\mathbb{E}_A\Big[R(A_S) - R_{emp}(A_S)\Big] \leq 2\beta_m + (4m\beta_m + B)\sqrt{\frac{\log\frac{1}{\delta}}{2m}} \tag{26}$$

where $R(A_S)$ is the generalisation error/risk and $R_{\text{emp}}(A_S)$ is the empirical error. Finally, Equation 25, and Equation 26 in conjunction with our result i.e. Equation 24, immediately yield the following proposition:

Proposition 1. *Let A_S be a one-layer GNN algorithm (of Equation 4) equipped with the graph convolutional filter $g(L_G)$ and trained on a dataset S for T iterations. Let the loss and activation functions be Lipschitz-continuous and smooth. Then the following expected generalisation gap holds with probability at least $1-\delta$, $\delta \in \{0,1\}$:*

$$\mathbb{E}_{\text{SGD}}\Big[R(A_S) - R_{\text{emp}}(A_S)\Big]$$

$$\leq \frac{1}{m}O\Big((\lambda_G^{max})^{2T}\Big) + \Big(O\Big((\lambda_G^{max})^{2T}\Big) + B\Big)\sqrt{\frac{\log\frac{1}{\delta}}{2m}} \tag{27}$$

where the expectation $\mathbb{E}_{SGD}$ is taken over the randomness inherent in SGD, m is the no. training samples, and B is a constant which depends on the loss function. Our proposition states that GNN trained with the Wasserstein loss enjoys the same generalisation error bound as the traditional GCN (trained with cross entropy). We establish that such models, which use filters with bounded eigenvalues independent of graph size, can satisfy the strong notion of uniform stability and thus is generalisable.

The above analysis is quite general and the error bounds can be easily established for DHN and existing hypergraph neural methods based on the discussion in Section 4.1.

5 Additional Experiments

5.1 Ablation Study

In this section, we compared our proposed one-layer DHN against deeper layers and without exploiting directed hyperedges.

Table 1. Statistics of datasets used in the experiments.

Dataset	**Type**	$\|\mathbf{V}\|$	$\|\mathbf{E}\|$	$\|\mathbf{E_d}\|$	**F**	**Avg. edge size**	**Max. edge size**
Cora	Co-authorship	2653	2591	12071	7	2.3 ± 1.9	29
DBLP	Co-authorship	22535	43413	117215	5	4.7 ± 6.1	143
Amazon	Recommendation	84893	166994	1081994	5	3.0 ± 3.0	187
ACM	Co-authorship	67057	25511	59884	6	2.4 ± 1.2	32
arXiv	Co-authorship	790790	1354752	6728683	7	4.0 ± 19.7	2832

Dataset	GCN	Soft-GCN	GAT	Soft-GAT	Simple-GCN	Soft-Simple-GCN
Cora	81.5	81.7	82.8	82.5	81	81.6
Citeseer	70.3	70.1	70.6	70.6	71.8	71.6
Pubmed	79	78.6	78.7	78.4	78.8	78.8

Table 2. Accuracy on traditional graph-based SSL datasets. The experimental setting is the same as in GCN [7] and GAT [11]. We used 10% of the labelled vertices as validation data to tune the hyperparameter η.

5.2 Varying Labelled Data and increasing F

Table 4 shows the results on arXiv for varying percentage of labelled data. As we can see, Soft-DHN achieves the best results across all percentages of labelled nodes.

We also analysed at what point our proposed methods (KL-DHN and Soft-DHN) recovered performance on the largest arXiv dataset with all nodes (100%) labelled. KL-DHN achieved a 100∗MSE of 7.34, and Soft-DHN achieved a 100∗MSE of 5.87 with all nodes labelled. Moreover, KL-DHN required around 72% labelled nodes to achieve a 100∗MSE of 7.34 while Soft-DHN required around 79% to achieve a 100∗MSE of 5.87. As we can observe, KL-DHN plateus earlier (requires 72% of the labelled nodes) than Soft-DHN (79%). But Soft-DHN performs much better than KL-DHN.

As another experiment we increased the number of research interests for authors on the largest arXiv dataset from $F = 7$ to $F = 25$. The results of our method and the baselines are shown in Table 5. As we can see our proposed Soft-DHN achieves the best performance in this setting too.

6 Details of hyperparameters

Inspired by the experimental setups of prior related works [7, 8], we tune hyperparameters *of all methods including all baselines* using the Cora co-authorship network dataset alone. The optimal hyperparameters are fixed and then used for all the other datasets. Table 6 shows the list of hyperparameters used in the datasets. Prior works [7, 8] have extensively performed tuning of hyperparameters such as hidden size, learning rate, etc and we fixed their reported optimal hyperparameters. We hyperparameterise the cost matrix (base metric of the

Table 3. Ablation study of our proposed Soft-DHN.

# **DHN layers**	# **GNN layers**(hops)	**Cora**	**DBLP**
2	2	7.68 ± 0.24	7.98 ± 0.27
2	1	7.64 ± 0.25	7.93 ± 0.28
2	0	7.69 ± 0.27	7.98 ± 0.22
1	0	5.64 ± 0.32	6.45 ± 0.38
1	1	5.41 ± 0.35	6.26 ± 0.32
1	2	$\mathbf{4.87 \pm 0.40}$	$\mathbf{5.65 \pm 0.42}$

Table 4. Results on ***arXiv*** dataset. 100∗ Mean squared error ± standard deviation (lower is better) over 10 different train-test splits.

Model	**1%**	**3%**	**5%**	**10%**	**20%**
Soft-HGNN	8.61 ± 0.49	8.46 ± 0.52	8.42 ± 0.43	8.28 ± 0.37	8.11 ± 0.45
Soft-HyperGCN	8.60 ± 0.47	8.44 ± 0.43	8.40 ± 0.41	8.29 ± 0.40	8.15 ± 0.41
KL-DHN	9.34 ± 0.32	9.25 ± 0.34	9.19 ± 0.35	9.03 ± 0.41	8.88 ± 0.34
Soft-DHN	$\mathbf{7.69 \pm 0.36}$	$\mathbf{7.51 \pm 0.37}$	$\mathbf{7.41 \pm 0.34}$	$\mathbf{7.22 \pm 0.39}$	$\mathbf{7.06 \pm 0.32}$

Wasserstein distance) as follows:

$$C = \begin{bmatrix} 1 & \eta & \eta & \dots & \eta & \eta \\ \eta & 1 & \eta & \dots & \eta & \eta \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \eta & \eta & \eta & \dots & \eta & 1 \end{bmatrix}$$

Table 7 shows the best results on the validation split of Cora (with optimal hyperaparameters). The training set had 140 vertices, the validation set 1000 vertices and the rest of the vertices were used to test the models. The results reported are after 200 epochs of training with a seed value of 598.

Table 5. Increasing the dimension of F. new $F = 25$

Model	100***MSE**
Soft-HGNN	24.7 ± 3.3
Soft-HyperGCN	24.5 ± 3.8
Soft-Hyper-SAGNN	23.5 ± 2.8
Soft-HNHN	23.9 ± 2.9
Soft-DHGCN	23.3 ± 2.6
KL-DHN	26.7 ± 2.5
Soft-DHN	22.3 ± 2.9

Table 6. List of hyperparameters used in the experiments. A set of values indicates that the corresponding hyperparameter is tuned from the set (on the validation split).

Hyperparameter	**Value(s)**	**Hyperparameter**	**Value(s)**
hidden size, h	$\{8, 16, 32, 64, 128\}$	ϵ (Sinkhorn)	$\{0.0001, 0.001, 0.01, 0.1, 1, 10\}$
# Sinkhorn iterations	100	learning rate, r	$\{0.0001, 0.001, 0.01, 0.1, 1, 10\}$
dropout	0.5	η	$\{0, 1, 2, \cdots, 40\}$
weight decay	5×10^{-4}	λ	$\{1, 0.5, 0.1, 0.05, 0.01, \cdots, 5 \times 10^{-7}, 10^{-7}\}$

7 Sources of the real-world datasets

Co-authorship data: All authors co-authoring a paper are in one hyperedge. We used the author data[2]to get the co-authorship hypergraph for cora. We manually constructed the DBLP dataset from Arnetminer[3].

7.1 Construction of the DBLP dataset

We downloaded the entire dblp data from https://aminer.org/lab-datasets/citation/DBLP-citation-Jan8.tar.bz [10]. The steps for constructing the dblp dataset used in the paper are as follows:

– We defined a set of 5 conference categories (histograms for the SSL task) as "algorithms", "database", "datamining", "intelligence", and "vision"
– For a total of 4304 venues in the entire dblp dataset we took papers from only a subset of venues from Wikipedia corresponding to the above 5 conferences

[2] https://people.cs.umass.edu/ mccallum/data.html
[3] https://aminer.org/lab-datasets/citation/DBLP-citation-Jan8.tar.bz

Table 7. Optimal hyperparameters on the validation set on Cora Co-authorship network. We found that $h = 16, r = 0.01, \epsilon = 0.1$ were optimal for all methods in the table.

Method	Optimal hyperparameters	Best MSE on validation set
KL-MLP	-	7.87
OT-MLP	$\eta = 31$	6.47
KLR-MLP	-	7.39
OTR-MLP	$\eta = 25,\ \lambda = 5 \times 10^{-3}$	4.86
KL-HGNN	-	6.98
KL-HyperGCN	-	7.03
Soft-HGNN	$\eta = 20$	3.24
Soft-HyperGCN	$\eta = 17$	4.02
Soft-Hyper-Atten	$\eta = 19$	3.16
Soft-Hyper-SAGNN	$\eta = 17$	3.09
Soft-HNHN	$\eta = 17$	2.95
Soft-DHGCN	$\eta = 18$	3.04
KL-DHN	-	6.34
Soft-DHN	$\eta = 19$	**2.67**

– From the venues of the above 5 conference categories, we got 22535 authors publishing at least two documents for a total of 43413
– We took the abstracts of all these 43413 documents, constructed a dictionary of the most frequent words (words with frequency more than 100) and this gave us a dictionary size of 1425
– We then extracted the 117215 citation links among these documents

7.2 Construction of the Amazon Office Product dataset

We downloaded the entire Amazon data [6, 9]. The steps for constructing the dataset used in the paper are as follows:

– We downloaded the office product ratings subset from the entire dataset
– We constructed a hypergraph of items with each hyperedge representing a user connecting all the items that they bought

- We removed hyperedges of size 1
- We connected a pair of hyperedges (bi-directional) if they had more than 1 item in common

7.3 Construction of the ACM dataset

We downloaded the entire ACM data from https://lfs.aminer.org/lab-datasets/citation/acm.v9.zip [10]. The steps for curating the dataset in the paper are as follows:

- Based on the number of papers published, we identified the six most popular venues: "Journal of Computational Physics", "IEEE Transactions on Pattern Analysis and Machine Intelligence", "Automatica (Journal of IFAC)", "IEEE Transactions on Information Theory", "Expert Systems with Apllications: An International Journal", and "IEEE Transactions on Computers"
- We then listed the set of all authors published in these venues (we got a total of 67057 authors).
- We finally obtained the citation relationships of all the documents co-authored by these authors (total number of documents is 25511 and total number of citations is 59884)

7.4 Details of the ArXiv dataset

We downloaded the entire arXiv dataset [2] from https://github.com/mattbierbaum/arxiv-public-datasets/releases/tag/v0.2.0. The steps for curating the dataset in the paper are as follows:

- We removed papers without any authors and got a total of $13,54,752$ edges
- We extracted $67,28,683$ citation edges among these papers
- The total number of authors in these papers is $7,90,790$

References

1. Altschuler, J., Weed, J., Rigollet, P.: Near-linear time approximation algorithms for optimal transport via sinkhorn iteration. In: NIPS. Curran Associates, Inc. (2017)
2. Clement, C.B., Bierbaum, M., O'Keeffe, K.P., Alemi, A.A.: On the use of arxiv as a dataset. CoRR, abs/1905.00075 (2019)
3. Cuturi, M.: Sinkhorn distances: Lightspeed computation of optimal transport. In: NIPS. Curran Associates, Inc. (2013)
4. Feng, Y., You, H., Zhang, Z., Ji, R., Gao, Y.: Hypergraph neural networks. In: AAAI (2019)
5. Hardt, M., Recht, B., Singer, Y.: Train faster, generalize better: Stability of stochastic gradient descent. In: ICML (2016)
6. He, R., McAuley, J.: Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In: WWW (2016)
7. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)

8. Liao, R., Zhao, Z., Urtasun, R., Zemel, R.S.: Lanczosnet: Multi-scale deep graph convolutional networks. In: ICLR (2019)
9. McAuley, J., Targett, C., Shi, Q., van den Hengel, A.: Image-based recommendations on styles and substitutes. In: SIGIR (2015)
10. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: Arnetminer: Extraction and mining of academic social networks. In: KDD (2008)
11. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: ICLR (2018)
12. Verma, S., Zhang, Z.L.: Stability and generalization of graph convolutional neural networks. In: KDD (2019)
13. Villani, C.: Topics in optimal transportation theory (2003)
14. Villani, C.: Optimal transport – Old and new, vol. 338. Springer-Verlag (2008)
15. Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., Weinberger, K.: Simplifying graph convolutional networks. In: ICML (2019)
16. Zhou, D., Huang, J., Schölkopf, B.: Learning with hypergraphs: Clustering, classification, and embedding. In: Schölkopf, B., Platt, J.C., Hoffman, T. (eds.) NIPS. MIT Press (2007)